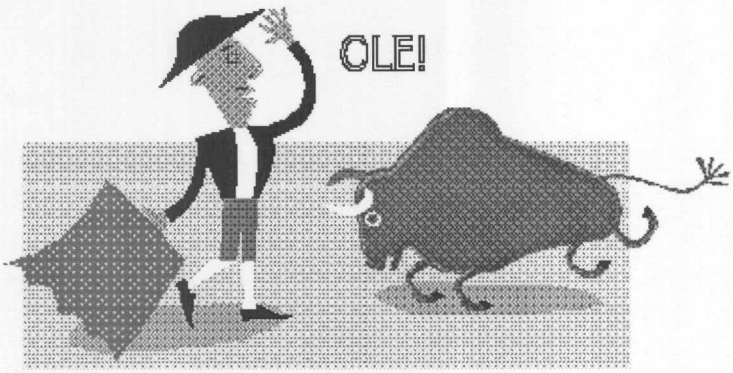
The Unofficial Newsletter of Delphi Users - Issue #21 - May 1997



Excel OLE Tips for Everyone:

by Joselito Real - reajos@kinwticsys.com

I mainly use Excel to dump the output of my database reports because of the very good printer output that I can get, and besides, my clients want all the output in Excel so that they can tweak the reports a little bit and still print it or reformat according to their whims.

To make my tips work, you must have at least Excel version 7, I haven't tried the earlier versions of Excel. I have been using OLE to drive Excel at a very acceptable speed for my clients, of course, if I know ALL about the Excel Type of Clipboard format to dump to Excel, that is the fastest way to go and dump my output there. However, OLE is here, and after digging into it, here are some things that I have uncovered by myself:

An example of program flow might be:

```
Procedure ExcelOut;
  var  XL:variant;
begin
 //start Excel by creating an instance of Excel Application
 //add a workbook where you dump your data
 //do your data dump routines
 //control Excel's cell format
 //control Excel's page setup
 //make Excel visible
 //tell excel to preview/print your report
 //tell Excel to save your output
 //quit and free the instance of Excel
end;
```

Here are some details:

```
  // start Excel by creating an instance of Excel Application

  XL:=CreateOLEObject('Excel.Application');

  // add a workbook where you dump your data
```

```
XL.WorkBooks.add;
```

You must include declaration of variant variables to facilitate your data dump, for example, included in the **var** statement following procedure ExcelOut;

```
// do your data dump routines

Var XL, XArr: Variant;
```

then after the begin statement you may wish to create XArr, ie.,

```
XArr:=VarArrayCreate([1,10],varVariant);
```

the array's purpose in our example is to dump 10 cells at a time per OLE call to Excel. Then load your data to XArr, for example:

```
XArr[1]:=12.85;
XArr[2]:='Yes';
XArr[3]:='California';
XArr[4]:=Table1.FieldByName('FIRSTNAME').AsString;
XArr[5]:='';
XArr[6]:=i;   //i must have been predeclared elsewhere
XArr[7]:=TempStr;  //TempStr is a string predeclared elsewhere
XArr[8]:=AVG;  //AVG is a predeclared real number
XArr[9]:=STD;  //STD is a predeclared real number
XArr[10]:=j;  //j is a predeclared integer
```

of course, it is always better to load the array using a for-next loop instead of the brute-force illustration above. A sample of such application is to transfer a record of a database table to an Excel row, ie.:

```
XArr:=VarArrayCreate([1,Table1.FieldDefs.Count],varVariant);
for i:=1 to Table1.FieldDefs.Count do
  XArr[i]:=Table1.Fields[i-1]; //first field of a table is 0
                               //FieldDefs.Count is no. of fields
```

whatever method you choose, you then dump XArr unto Excel for example, we want to dump XArr to first row and from columns A to J:

```
XL.Range('A1:J1').Value:=XArr;
```

another programmatic approach may be to define RowRange as string and then assign string values to RowRange controlled by our program, ie.:

```
RowRange:='A'+IntToStr(j)+':'+CHR(64+Table1.FieldDefs.Count)+IntToStr(j);
XL.Range(RowRange).Value:=XArr;
```

of course, as long as the number of fields do not go over 26 columns for the above example, where j in

the above column is the desired row number in the excel spreadsheet. You may also dump a 2-dimensional array at a time but please avoid dumping one cell at a time because OLE process could become sooo slooowwww!

Assuming you have dumped your data, you may wish to format them, like add bold lines, thick lines, adjust to correct cell width, and here are the list of Excel OLE commands from within Delphi:

To format a cell or group of cells, you must select it first:

```
//control Excel's cell format

XL.Range('A1:J25').Select;
```

To select the entire spreadsheet cells use:

```
XL.cells.select;
```

Then apply the format commands:

```
XL.Selection.Font.Name:='Arial Cyr';
XL.Selection.Font.Size:=9;
XL.selection.Columns.AutoFit;
```

And here some other detailed cell formatting commands: These commands will generate a border for the selected cells using thin lines. For other types of lines, you may email me on how to get them

```
    XL.Selection.Borders(xlLeft).Weight := xlThin;
    XL.Selection.Borders(xlRight).Weight := xlThin;
    XL.Selection.Borders(xlTop).Weight := xlThin;
    XL.Selection.Borders(xlBottom).Weight := xlThin;
```

For these detailed commands to work, you must predeclare:

```
const
  xlLeft=-4131;
  xlRight=-4152;
  xlTop=-4160;
  xlBottom=-4107;
  xlThin=2;
  xlHairline=1;
  xlNone=-4142;
  xlAutomatic=-4105;
```

where did I get the values for these constants? email me!

```
  //control Excel's page setup

  XL.ActiveSheet.PageSetup.PrintTitleRows := 'A1:J1'; //Repeat this row/page
  XL.ActiveSheet.PageSetup.LeftMargin:=18;   //0.25" Left Margin
```

```
XL.ActiveSheet.PageSetup.RightMargin:=18;  //0.25"  will vary between printers
XL.ActiveSheet.PageSetup.TopMargin:=36;    //0.5"
XL.ActiveSheet.PageSetup.BottomMargin:=36; //0.5"
XL.ActiveSheet.PageSetup.HeaderMargin:=18; //0.25"
XL.ActiveSheet.PageSetup.FooterMargin:=18; //0.25" Footer Margin
XL.ActiveSheet.PageSetup.CenterHorizontally:=1; //zero, means not centered
XL.ActiveSheet.PageSetup.Orientation:=2; //landscape=2, portrait=1

//make Excel visible
XL.visible:=true;

//tell excel to preview/print your report
XL.ActiveSheet.PrintPreview;  //for previewing
XL.ActiveWindow.SelectedSheets.PrintOut (Copies := 1); //print directly

//tell Excel to save your output
XL.ActiveWorkBook.SaveAs ('MyOutput');
```

you can also save your output in other formats

```
//quit and free the instance of Excel

XL.visible:=False;
XL.quit;
XL:=unassigned;
```

How Did I Dig Out These Commands?

Well, it is easy, just run your Excel application, record a macro, and then do what you want, like page setup, open file, save file, sort, etc,... then stop recording the macro, then print your recorded macro, LO and behold!

All the commands are exposed for your perusal in Delphi, just tweak them to conform to Delphi Pascal's syntax.

My Best Tip for the Fastest Transfer of Database Data to Excel Using OLE:

Batchmove your query results, paradox tables, or sections of your large dBASE Tables into a temporary Table having a format of dBASEIV or earlier format and then using OLE, control Excel to convert the table instantly into Excel format or print the Table in Excel by opening your temporary table as a dBASE file.

For example, Temp.DBF is a dBASEIV formatted file produced as a result of a TBatchMove procedure from a query. To start the process of conversion to Excel file do the following:

```
Procedure ExcelOut;
  var  XL:variant;
begin
  XL:=CreateOLEObject('Excel.Application');
  XL.workbooks.open ('\Temp.DBF');  //supply the directory path if needed
  XL.ActiveWorkBook.SaveAs (Filename := '\MyTable', FileFormat :=  -4143);
  //the above line saves whatever loaded file as MyTable.XLS
  //at this point, conversion to Excel File is done, no need for further code
```

```
    //except for cleaning up the instance of Excel Application
    //but say, you want a nice printed output, add the following codes:
    XL.Cells.select; //Select all Cells and prepare for format
    XL.Selection.Font.Name:='Arial';
    XL.Selection.Font.Size:=9;
    XL.selection.Columns.AutoFit;
    XL.ActiveSheet.PageSetup.PrintTitleRows := '$1:$1'; //repeat column headings/pa
    XL.ActiveSheet.PageSetup.PrintGridlines := 1;       //print with grid lines
    XL.ActiveWindow.SelectedSheets.PrintOut (Copies := 1); //print directly to
    //House-cleaning is required
    XL.quit;
    XL:=unassigned;
  end;
```

The above example, is the fastest so far (less than 10 seconds on my Pentium 133 MHz machine to transfer 15 Fields by 9,656 records!) to convert your medium to large dBASE tables into Excel using only the current OLE technology but without using those expensive conversion DLL's or OCX's, and without using those QuickReports and other fancy printer formatting tools!

Well I hope you have some fun in using my tips! You can even build a faster OLEExcel component using my tips! Should you find my tip useful, kindly post my dream component in the appropriate places:

One of my wishes is really that if someone out there could write an XLDBGrid or XLStringGrid component:

This component should have the ease of use of cut and paste type of data entry for repetitive data as you would with an Excel spreadsheet, the use of ctrl-C, ctrl-V, Ctrl-X, direct cut and paste to/from an open Excel spreadsheet onto these Grid components are allowed, it should also facilitate the ease of inserting/deleting cells or entire rows or columns, it should have the same way of selecting the group of cells using a mouse or shift keys. No need to include the formula computations. You see, almost everybody who have some exposure to computers knows how to enter data in an Excel spreadsheet without the need for further training. Let them enter Tabular data in a DBGrid... AAARRRRGH!

The Paradox style DBGrid in Delphi is such a pain to enter your data, for example all the other columns are correct, except that there is one item you need to insert into that one-column and everything will be aligned correctly--you don't need to reenter or type over those other entries--just move them automatically after inserting an item! The same is true for the DBEdits, they are not suited for tabular types of data. Most receipts and scientific data are tabular, and you enter them in a tabular format, with the ease and convenience of an Excel or Lotus spreadsheet! If such a component is available, without me doing the code for those mouse and key controls, and without digging those Microsoft Clipboard Excel format, it would be a very nice front end for tabular types of data entry all controlled within Delphi. Yes there is a Formula One OCX, that is available but it is an overkill! and besides, you have to intercept or redefine the way it is handling the clipboard. Lots of work to do to emulate it to behave like an Excel-style of data entry and the OCX that ships together with your program is a big file. So please, if somebody out there who has this type of StringGrid or DBGrid, I'd gladly buy that component. For me, it would be an indispensable data entry front-end.

Return to Tips & Tricks

Return to Front Page